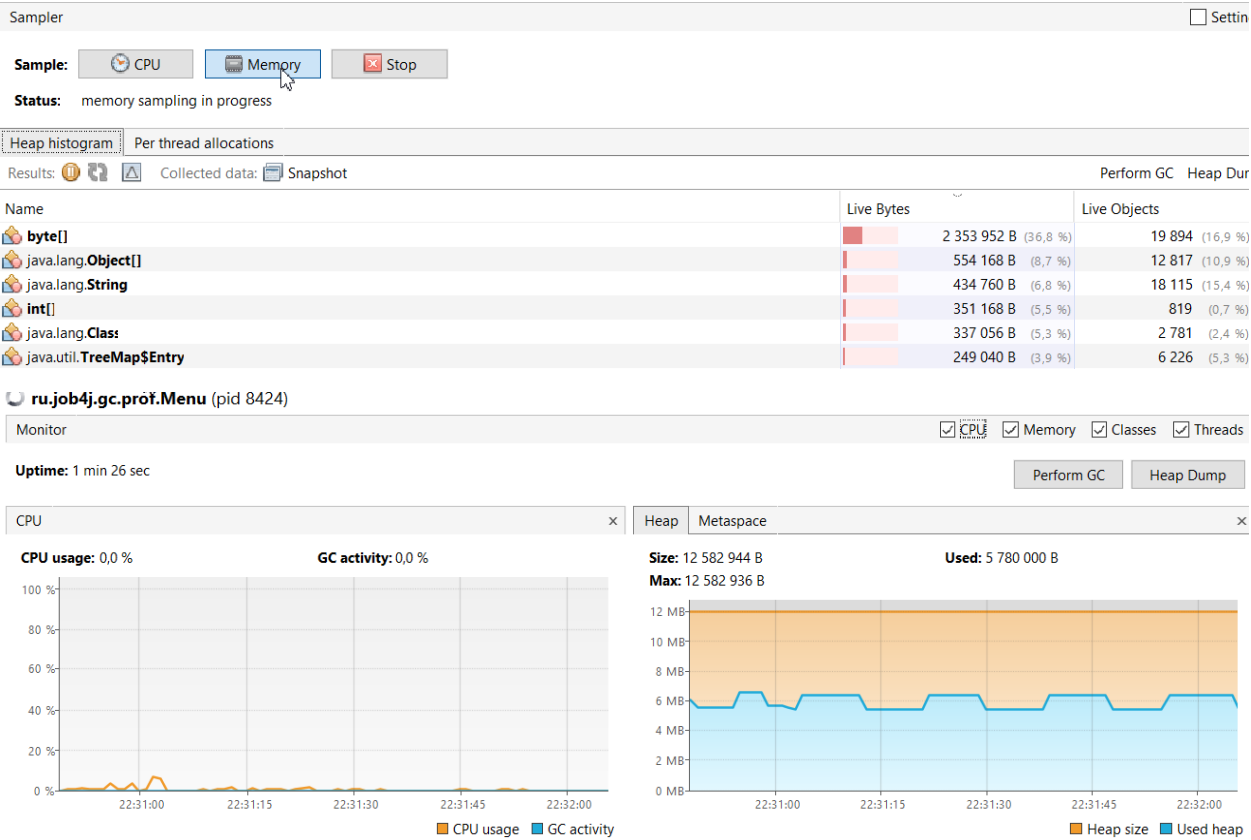
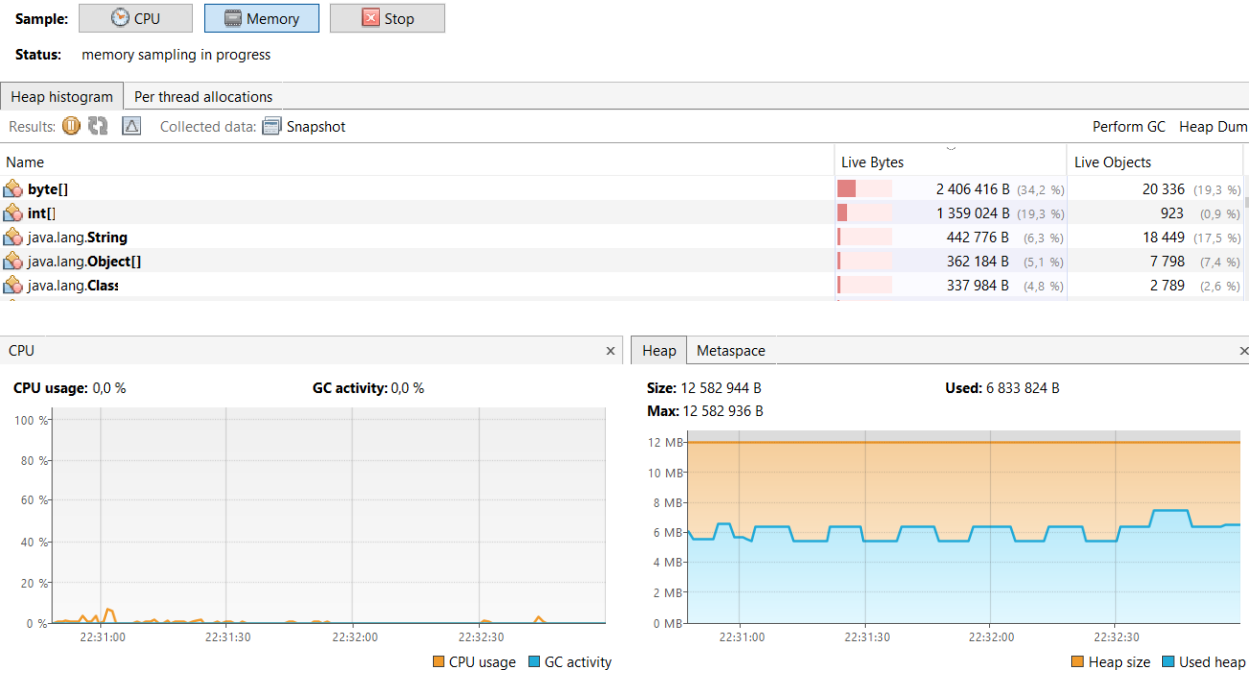


Начальный график отличается от Parallel и напоминает скорее график цифрового сигнала, без резких скачков. Видимо такое “сглаживание” результат многопоточного выполнения (Concurrent Mark Cycle). паузы судя по логам очень короткие

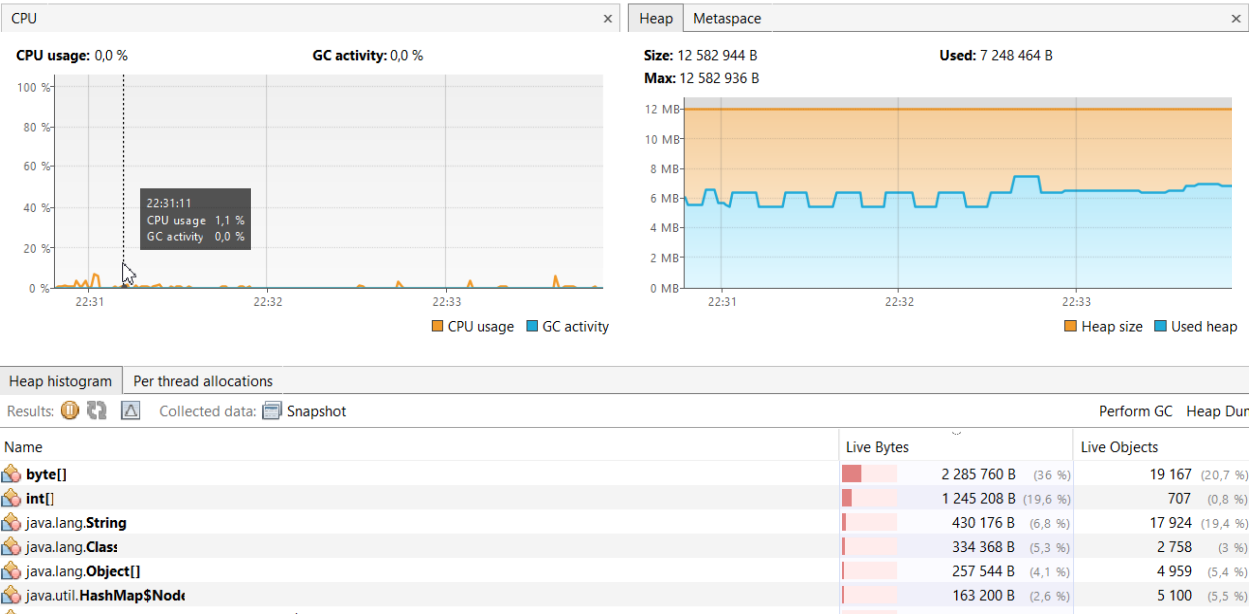


На создание 250000 также было выделено ~ 1_000_000байт, практически никак не загрузило CPU , и график немного отклонился в район ~ с 6,5 до 7,5 мб

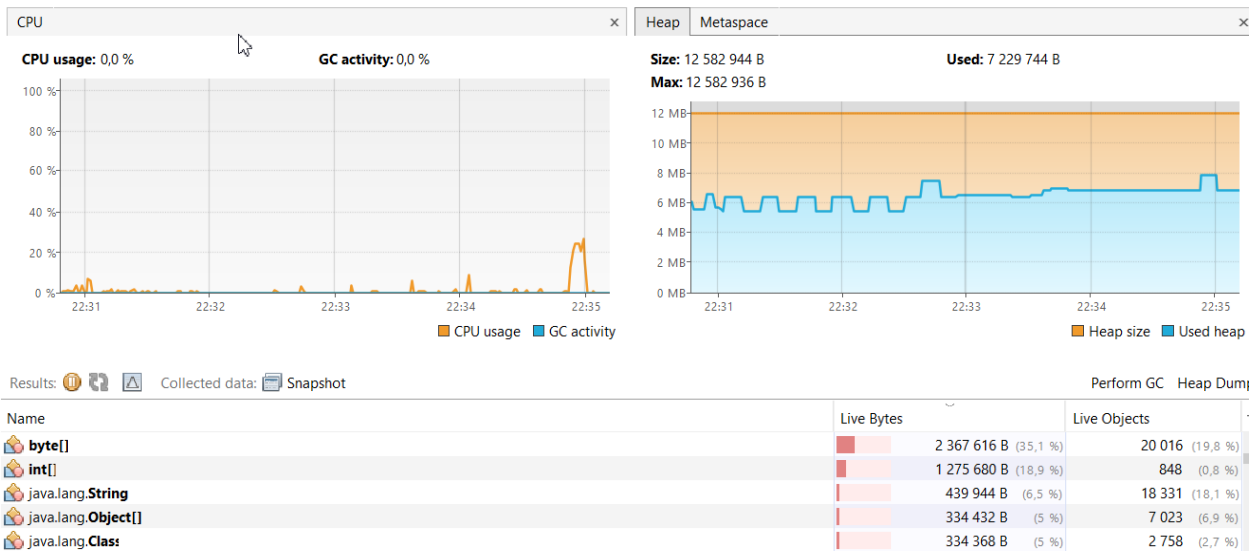


Сортировка слиянием выполнялась очень быстро ~12мс без особой нагрузки.

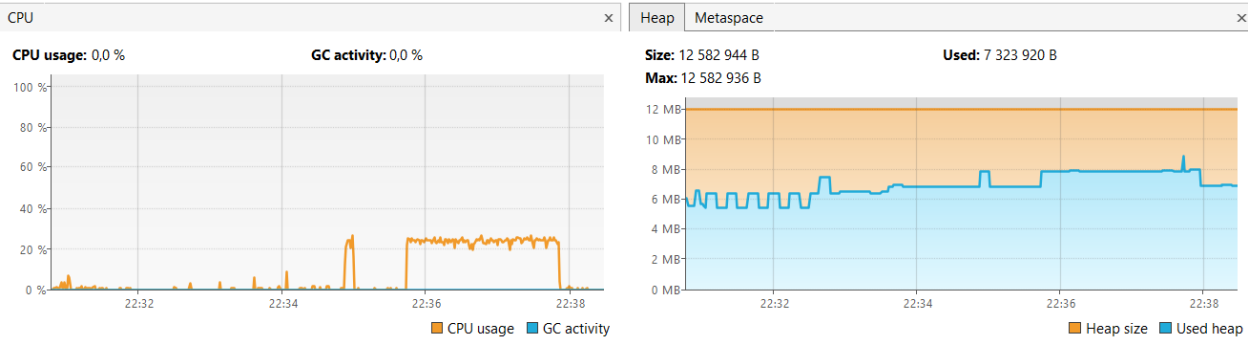
На графике не видно заметных выделений памяти



Сортировка вставками потребовала уже более заметных ресурсов, кратковременно выделилась память под int[], на что ушло примерно 7 секунд



Сортировка пузырьком была произведена примерно за 1м 45 сек и потребовала в разы больше ресурсов



Sample: CPU Memory Stop

Status: memory sampling in progress

Heap histogramPer thread allocations

Results: Collected data: Snapshot

Perform GCHeap Dump

Name	Live Bytes	Live Objects
byte[]	2 379 960 B (30,4 %)	20 177 (19,8 %)
int[]	2 323 824 B (29,7 %)	867 (0,9 %)
java.lang.String	439 800 B (5,6 %)	18 325 (18 %)
java.lang.Object[]	334 768 B (4,3 %)	7 016 (6,9 %)
java.lang.Class	334 368 B (4,3 %)	2 758 (2,7 %)
java.util.HashMap\$Node	165 216 B (2,1 %)	5 163 (5,1 %)
char[]	139 160 B (1,8 %)	448 (0,4 %)

Sampler

Sample: CPU Memory Stop

Status: memory sampling in progress

Heap histogramPer thread allocations

Results: Collected data: Snapshot

Perform GCHeap Dump

Name	Live Bytes	Live Objects
byte[]	2 299 456 B (36,2 %)	19 488 (2,1 %)
int[]	1 249 128 B (19,6 %)	710 (0,8 %)
java.lang.String	436 104 B (6,9 %)	18 171 (19,6 %)
java.lang.Class	334 144 B (5,3 %)	2 756 (3 %)
java.lang.Object[]	248 288 B (3,9 %)	4 730 (5,1 %)

Большая часть графика характерна ровными участками и тк особенностью данного сборщика является высокая пропускная способность – сборка проводится редко

